# Part 1

## 1.1. Give weights after training on each data point (3 total weight changes, one after each timestep/data point).

| | w1 | w2 | b |
|---|---|---|---|
| **First Step** | 0.2 | 0.1 | 0.1 |
| **Second Step** | 0.071 | -0.288 | -0.029 |
| **Third Step** | 0.071 | -0.476 | -0.076 |

## 1.2. Show your work in calculating the values of the weights after training on each data point.

We start out with w1, w2, and b initialized at 0, and a learning rate of 0.2.
Our update rule is gonna be:
w1 = w1 - 0.2 * error * x1
w2 = w2 - 0.2 * error * x2
b = b - 0.2 * error

Then it's just a matter of processing each datapoint.

1. FIRST (x1=2, x2=1, y=1)
   z = w1*x1 + w2*x2 + b = 0*2 + 0*1 + 0 = **0**
   error = 1/(1+e^-z)-1 = 1/(1+e^0)-1 = 1/2-1 = **-0.5**
   update:
      w1 = 0 - 0.2 * -0.5 * 2 = **0.2**
      w2 = 0 - 0.2 * -0.5 * 1 = **0.1**
      b = 0 - 0.2 * -0.5 = **0.1**
2. SECOND (x1=1, x2=3, y=0)
   z = w1*x1 + w2*x2 + b = 0.2*1 + 0.1*3 + 0.1 = **0.6**
   error = 1/(1+e^-z)-0 = 1/(1+e^-0.6)-0 = 1/1.549-0 = **0.646**
   update:
      w1 = 0.2 - 0.2 * 0.646 * 1 = **0.071**
      w2 = 0.1 - 0.2 * 0.646 * 3 = **-0.288**
      b = 0.1 - 0.2 * 0.646 = **-0.029**
3. THIRD (x1=0, x2=4, y=0)
   z = w1*x1 + w2*x2 + b = 0.071*0 + -0.288*4 + -0.029 = **-1.181**
   error = 1/(1+e^-z)-0 = 1/(1+e^1.181)-0 = 1/1.307-0 = **0.235**
   update:

w1 = 0.071 - 0.2 * 0.235 * 0 = **0.071**
w2 = -0.288 - 0.2 * 0.235 * 4 = **-0.476**
b = -0.029 - 0.2 * 0.235 = **-0.076**

### 1.3. Briefly comment on any shift in weights from positive to negative or negative to positive and why this was the case.

w1: Remained positive. I'd say x1 contributed positively at first (2), then didn't do much to reduce it (1), and finally stabilized (0).

w2: Shifted from positive to negative. Initially increased like w1 (1), but the second and third data points led to the gradient pushing it towards negative because the values were too high (3 & 4).

b: Shifted from positive to negative. Started increasing, but since the second datapoint was classified as 0 but got predicted higher (0.645), the bias had to be shifted downward to correct for this.

# Part 2

## 2.1 Logistic Regression Models

Table of 5-Fold Cross Validation Scores

| Logistic Regression Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Bag of Words Unigram | 60.46% | 62.23% | 53.26% | 57.39% |
| Bag of Words w/ Selected Features | **65.35%** | **68.93%** | 55.89% | **61.73%** |
| Character-Level TF-IDF | 60.65% | 60.81% | **59.90%** | 60.35% |

**Motivation & Results for the BoW w/ Selected Features:**
For the first altered model, my thinking was that if I implemented some chi-squared feature selection to identify the 1000 most discriminative words that I could turn focus away from misleading and irrelevant vocabulary. This ended up increasing performance somewhat across the board, and resulted in the best model of the regression attempts.

**Motivation & Results for the BoW w/ Selected Features:**
This really didn't do too well, but several of my other attempts actually dropped reliability towards the realm of a coin toss. I guess I was hoping that some more subtle writing patterns could emerge from proximal characters and guide the model towards something more consistent. And if consistency is all I was after, I guess this model did have a pretty good precision/recall trade-off?

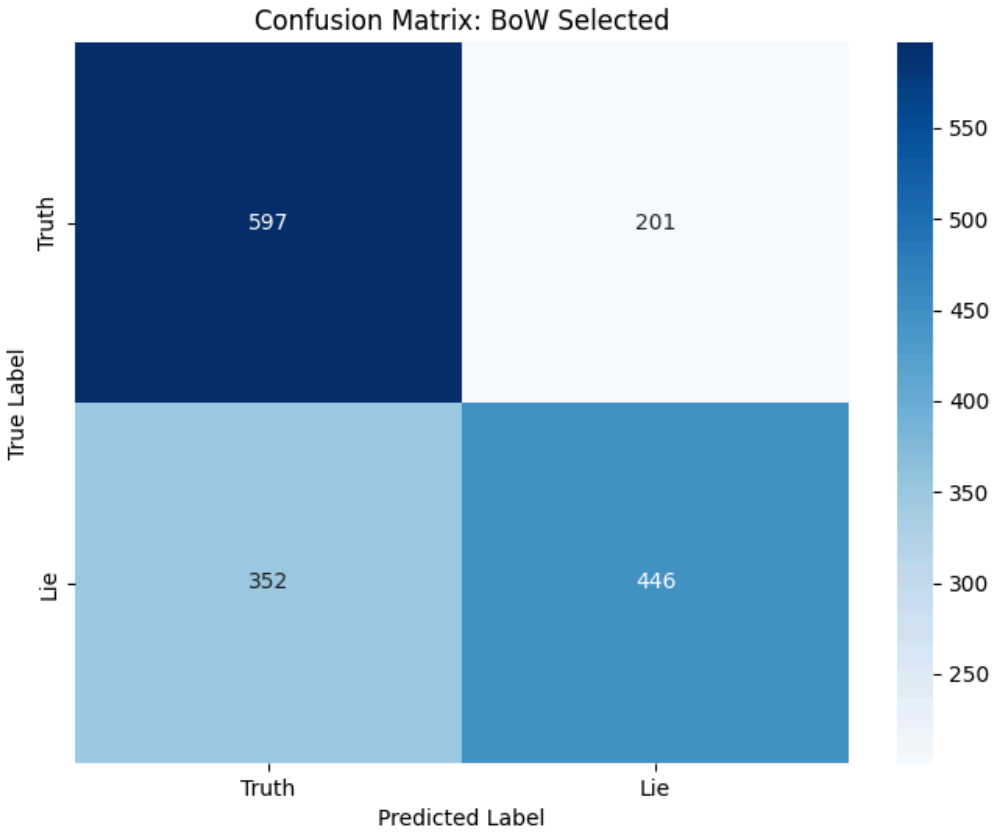
**BoW w/ Selected Features - Feature Analysis:**
Top positive features:
okay: 1.5533
alright: 1.4599
since: 1.3090
fall: 1.3064
attack: 1.1601

Top negative features:
dude: -1.1840
supported: -1.0650
med: -1.0443
shot: -1.0411
year: -1.0363

I wasn't sure what to think about some of these at first. "Okay" and "Alright" come across as fake reassurances, and "Since" feels like it would be woven into someone's deceptive reasoning toolbox. "Fall" and "Attack" I'm not as sure on, maybe fear mongering? So-and-so will attack you, your __ will fall, etc.?

For the negative features, I really would have thought "Dude" would be an indicator of lies. At least in the social deception games I've played, anyone dropping a "dude" is getting found out for their deceit. "Supported" probably implies a focus on cooperative progression and agreements that would not be as high on the vocabulary of someone trying to stoke fear and control a narrative, but honestly… I'm curious about that. Not like this model did all that well. "Med", "Shot", and "Year" I can't even begin to muster a guess on.

**BoW w/ Selected Features - Error Analysis:**




[x] False Positive Examples (Predicted lie, actually truth):
[>] Text: Yes- I would very much love to see that. And the best part is that you get to see my super friendly ...
[>] Text: What I was/am trying to avoid is a scenario in which you (a) work with Germany and (b) send way more...
[>] Text: ...okay, clearly I don't have your trust. But really, I am committed to an FG victory! I'm convoying...

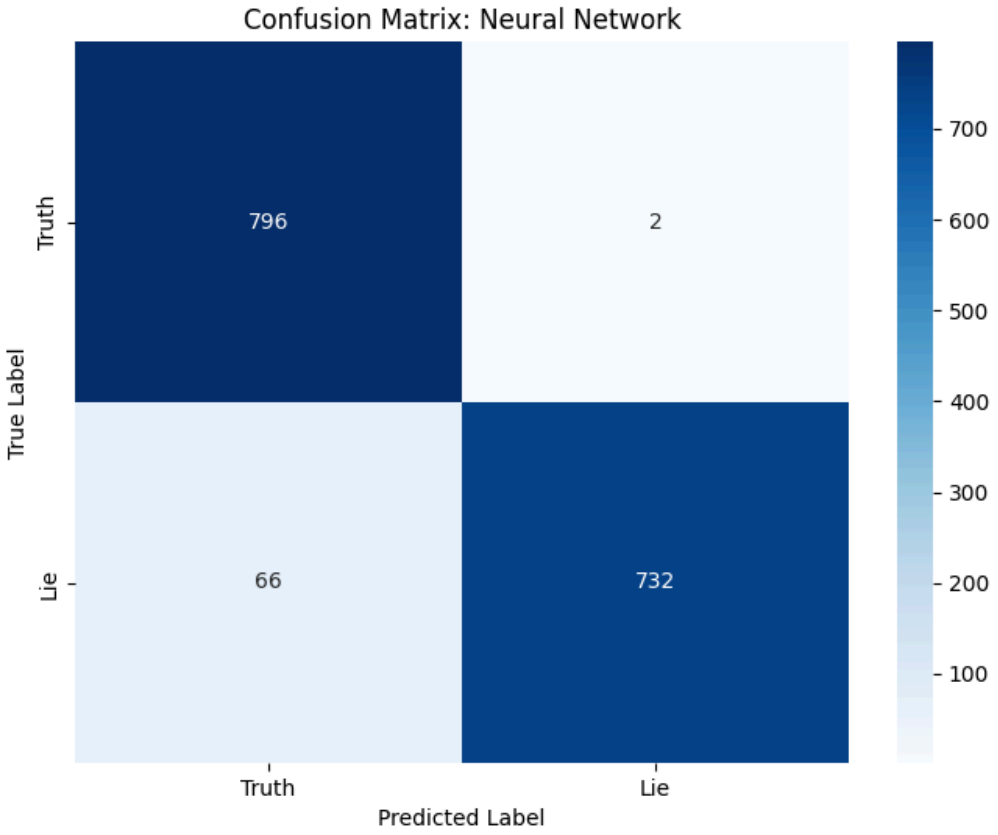
[x] False Negative Examples (Predicted truth, actually lie):
[>] Text: Done...
[>] Text: Updated moves...
[>] Text: Don't believe Turkey, I said nothing of the sort. I imagine he's just trying to cause an upset betwe...

Okay, so let's see. I've been sifting through examples, but there are not that many particularly good ones. I *can* see how the False Positives come across as seeming as though they're beating around the bush or being overly wordy, behaviors I could see being classified as deceptive. For the False Negatives, I notice a lot of very short statements would show up as I reran the collection. The third example I copied above is a great example, in my opinion, of the "mud-flinging" that liars engage in when in a deceptive game.

As for how to improve things, I definitely think that building something to more intelligently track individual conversation threads and trace them throughout the games could make it easier to improve classification of short phrases and help tilt things one way or the other with longer sentences. Negation and contradiction detection (“I said nothing of the sort!”) and any other telltale signs of deception could also somehow be encoded and worked into the models for a boost.

## 2.2 Neural Network

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Bag of Words w/ Selected Features | 65.35% | 68.93% | 55.89% | 61.73% |
| Neural Network | **95.74%** | **99.73%** | **91.73%** | **95.56%** |




My Neural Network performance improved a bit over using word embeddings when I decided to use the BoW Selected Features model I’d already trained. The dataset is a bit small, I’d imagine, for effective fine-tuning of pre-trained embeddings. I went with steadily decreasing

layer sizes to help gradually compress the feature space.
Layer 1: Linear(input_dim -> 256) + BatchNorm + LeakyReLU + Dropout(0.3)
Layer 2: Linear(256 -> 128) + BatchNorm + LeakyReLU + Dropout(0.2)
Layer 3: Linear(128 -> 2)

I resorted to consulting with an LLM and my data scientist partner to go over what some of these meant and how they contribute, and it is all in need of a further look by me, because I definitely would not be able to intelligently quote myself on any of this. BatchNorm layers help with training stability and speeding up convergence, LeakyReLU was chosen over base ReLU to help prevent “dying neurons” and wasted effort, and the decreasing dropout rates help ensure that more information is maintained in deeper layers. A learning rate of 0.001 was chosen as it seemed standard in a lot of examples I was reviewing. Batch sizes of 32 allow for enough batches per epoch to hopefully enforce meaningful updates, and after some serious struggles, it was advised that I set a patience-threshold of 5 to help prevent overfitting.

I experimented with a variety of settings and changes, but settled on this model in the end. I probably could have gone out to find some pre-trained word embeddings or fine-tuned BERT embeddings to see how differently the results would have played out. Varying the number of layers and their sizes, and learning more about the *types* of layers, would be a very good step for me to take if I wasn’t rushing to get ready for a conference.

I am fascinated by neural nets, and there’s obviously a whole world of techniques here for me to dive into. Hopefully I’ll get the chance with my semester project! Thanks again for your time and effort!